Amendments to the Claims

1. **(Currently Amended)** A method of representing type information for a typed intermediate language via objects of classes in a class hierarchy, wherein the class hierarchy comprises at least one class and a plurality of sub-classes for representing different type classifications, the method comprising:

instantiating one or more objects of one or more of the sub-classes of the hierarchy, wherein the one or more sub-classes represent classifications of types for the typed intermediate language; and

storing information in the one or more objects;

wherein the typed intermediate language is capable of representing a plurality of different programming languages, and wherein the one or more objects represent type information for instructions in the typed intermediate language; ~~and~~

wherein the classifications of types comprises a primitive type associated with a primitive type size, and wherein the primitive type size is settable to represent a constant size, the primitive type size is settable to represent a symbolic size, and the primitive type size is settable to represent an unknown size**; and**

wherein one of the sub-classes representing a primitive type represents an unknown type, wherein the unknown type can represent any type, and wherein a compiler drops type information by changing a known type to the unknown type during a stage of lowering.

2. (Canceled)

3. (Original) The method of claim 1 wherein at least one of the objects comprises information for a size of a type represented by the object.

4. (Previously Presented) The method of claim 1 wherein at least one of the one or more sub-classes inherits from an abstract type that wraps an externally defined type, the abstract type providing a mapping from the typed intermediate language to original source code.

5. (Original) The method of claim 1 wherein at least one of the one or more subclasses represents container types.

6. (Original) The method of claim 1 wherein at least one of the one or more subclasses represents pointer types.

7. (Original) The method of claim 1 wherein at least one of the one or more subclasses represents function types.

8. (Original) The method of claim 1 wherein at least one of the one or more subclasses represents unmanaged array types.

9. (Original) The method of claim 1 wherein at least one of the one or more subclasses represents class types.

10. (Original) The method of claim 1 wherein at least one of the one or more subclasses represents managed array types.

11. (Original) The method of claim 1 wherein at least one of the one or more subclasses represents struct types.

12. (Original) The method of claim 1 wherein at least one of the one or more subclasses represents interface types.

13. (Original) The method of claim 1 wherein at least one of the one or more subclasses represents enumerated types.

14. (Canceled)

15. (Previously Presented) The method of claim 1 wherein at least one of the subclasses representing primitive types represents the following types: int, float, and void.

16. **(Canceled)**

17. (Previously Presented) The method of claim 1 wherein at least one of the sub-classes representing primitive types is extensible to represent one or more additional primitive types.

18. (Original) The method of claim 1 wherein at least one of the one or more sub-classes is defined from the group consisting of: 'ContainerType', 'PtrType', 'FuncType', 'ClassType', 'StructType', 'InterfaceType', and 'EnumType'.

19. (Original) The method of claim 1 wherein at least one of the one or more sub-classes is defined as 'PrimType'.

20. (**Currently Amended**) A computer-readable medium **storing** ~~having~~ a software program thereon, the program comprising computer executable instructions for implementing a method for representing type information for a typed intermediate language using a class hierarchy for representing different type classifications, the method comprising:

defining a programming class of the class hierarchy as 'PrimType', wherein the programming class represents primitive type information for the typed intermediate language;

associating a size with instances of the 'PrimType' class, wherein the size is settable to represent an actual size of instances of the 'PrimType' class, settable to represent a symbolic size of instances of the 'PrimType' class, and settable to represent an unknown size of instances of the 'PrimType' class**, and wherein the actual size and the symbolic size are defined as a number of bits**; and

associating a kind of type with instances of the 'PrimType' class**;**

wherein the class 'PrimType' represents a plurality of types comprising at least an unknown type, wherein the unknown type can represent any type, and wherein a compiler drops type information by changing a known type to the unknown type during a stage of lowering.

21. (Original) The computer-readable medium of claim 20 wherein the size represents a size of a machine representation of a value.

22. **(Canceled)**

23. (Original) The computer-readable medium of claim 20 wherein the kind of type represents a type classification.

24. (Previously Presented) The computer-readable medium of claim 20 wherein associating a kind of primitive type with instances of the 'PrimType' class comprises defining the kind of type as 'PrimTypekind'.

25. (Canceled)

26. (Previously Presented) The computer-readable medium of claim 20 wherein associating a type of size with instances of the 'PrimType' class comprises defining the type of size as 'SizeKind'.

27. (Canceled)

28. **(Currently Amended)** The computer-readable medium of claim 20 wherein the class 'PrimType' **further** represents ~~**a plurality of types, the plurality of types comprising**~~ int, float, ~~**unknown,**~~ void, condition code, and unsigned int types.

29-30. (Canceled)

31. **(Currently Amended)** A method for representing type information for a typed intermediate language using a class hierarchy by programmatically defining a type representation, the method comprising:

defining a base class of the class hierarchy;

defining a plurality of classes hierarchically below the base class, wherein the plurality of classes represent type information for the typed intermediate language, and wherein the plurality of classes represent at least pointer types, container types and function types of a plurality of programming languages, and wherein the plurality of classes further comprise primitive types and the primitive types are associated with a primitive type size settable to represent a constant size, settable to represent a symbolic size, and settable to represent an unknown size;

wherein one of the primitive types represents an unknown type, wherein the unknown type can represent any type, and wherein a compiler drops type information by changing a known type to the unknown type during a stage of lowering.

32. (Previously Presented) The method of claim 31 further comprising defining a plurality of classes hierarchically below the class representing container types, wherein the plurality of classes represent type information for the typed intermediate language, and wherein the plurality of classes represent at least class types, struct types, interface types, and enumerated types of a plurality of programming languages.

33. (Previously Presented) The method of claim 32 further comprising defining a class hierarchically below the class representing class types, wherein the class represents type information for the typed intermediate language, and wherein the class represents unmanaged array types of a plurality of programming languages.

34. (Previously Presented) The method of claim 31 further comprising defining a class hierarchically below one of the plurality of classes, wherein the class represents type information for the typed intermediate language.

35. **(Canceled)**

36. **(Currently Amended)** A computer-readable medium **storing** ~~**having**~~ a software program thereon, the program comprising computer executable instructions for implementing a method for representing type information for a typed intermediate language using a class hierarchy for representing different type classifications, the method comprising:

defining a programming class of the class hierarchy as 'ContainerType', wherein an object of class 'ContainerType' is a type representation for the typed intermediate language for container types in a section of code written in one of a plurality of programming languages;

defining a programming class of the class hierarchy as 'PtrType', wherein an object of class 'PtrType' is a type representation for the typed intermediate language for pointer types in a section of code written in one of a plurality of programming languages;

defining a programming class of the class hierarchy as 'FuncType', wherein an object of class 'FuncType' is a type representation for the typed intermediate language for function types in a section of code written in one of a plurality of programming languages;

defining a programming class of the class hierarchy as 'ClassType', wherein an object of class 'ClassType' is a type representation for the typed intermediate language for class types in a section of code written in one of a plurality of programming languages;

defining a programming class of the class hierarchy as 'StructType', wherein an object of class 'StructType' is a type representation for the typed intermediate language for struct types in a section of code written in one of a plurality of programming languages;

defining a programming class of the class hierarchy as 'InterfaceType', wherein an object of class 'InterfaceType' is a type representation for the typed intermediate language for interface types in a section of code written in one of a plurality of programming languages;

defining a programming class of the class hierarchy as 'EnumType', wherein an object of class 'EnumType' is a type representation for the typed intermediate language for enumerated types in a section of code written in one of a plurality of programming languages; and

defining a programming class of the class hierarchy as 'PrimType', wherein an object of class 'PrimType' is a type representation for the typed intermediate language for primitive types in a section of code written in one of a plurality of programming languages;

wherein the object of class 'PrimType' is associated with a size settable to represent a constant size for the object of class 'PrimType', settable to represent a symbolic size for the object of class 'PrimType', and settable to represent an unknown size for the object of class 'PrimType'**; and**

wherein the class 'PrimType' represents a plurality of types comprising at least an unknown type, wherein the unknown type can represent any type, and wherein a compiler

drops type information by changing a known type to the unknown type during a stage of lowering.

37. (Canceled)

38. **(Currently Amended)** The computer-readable medium **storing** ~~**having**~~ a software program thereon, the program comprising computer executable instructions for implementing a method, of claim 36 wherein the method further comprises program code for associating a size with an object of any class.

39. **(Currently Amended)** The computer-readable medium **storing** ~~**having**~~ a software program thereon, the program comprising computer executable instructions for implementing a method, of claim 36 wherein the method further comprises program code for associating a kind of type with an object of any class.